AD-A239 228

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

...ated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, ... reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this ... burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson ... Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

...ORT DATE

3. REPORT TYPE AND DATES COVERED
FINAL, 01 NOV 89 to 31 DEC 90

4. TITLE AND SUBTITLE
PARALLELLOGIC PROGRAMMMING AND PARALLEL SYSTEM SOFTWARE AND HARDWARE

5. FUNDING NUMBERS
AFOSR-90-0027
61102F 2304/A7

6. AUTHOR(S)
Dr. Minker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Univeristy of Maryland
Computer Science Department
College Parkm MD 20742

8. PERFORMING ORGANIZATION REPORT NUMBER
91 0689

DTIC AUG 1991

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-90-0027
61102F 2304/A7

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This constitutes the final report of work performed under AFOSR grant number 90-0027 to investigate parallel problem solving and deductive databases. Under the grant experiments were performed on the PRISM parallel inference system on the BBN Butterfly. The experiments evaluated alternative message passing strategies for distributing tasks to processors at run-time. Several enhancements were made to PRISM during the grant period. These are: a new inference engine was implemented which provides more effieienct support for the full control language of PRISM; and a stack based inference engine was implemented which provides efficient support for the use of limited set of control strategies. Simulation studies were performed which evaluate alternative methods for scheduling tasks on parallel architectures. Two methods were examined which allow thek OR-parallel execution of logic programs with no communication overhead. A study was perfomred evaluating two alternative methods for incorporating integrity constraints into query processing in PRISM. In the first method, separate constraint processors are introduced which check constraints at run-time. In the second method, contraints are incorporated through compile-time transformations. The study indicates that constraints are useful inquery processing and that the compile-time methodology results in more efficient performance than checking constraints at run-time. In addition to the above, work continued in the area of informative answers to queries in deductive databases.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

Abstract

This constitutes the final report of work performed under AFOSR grant number 90-0027 to investigate parallel problem solving and deductive databases. Under the grant experiments were performed on the PRISM parallel inference system on the BBN Butterfly. The experiments evaluated alternative message passing strategies for distributing tasks to processors at run-time. Several enhancements were made to PRISM during the grant period. These are: a new inference engine was implemented which provides more efficient support for the full control language of PRISM; and a stack based inference engine was implemented which provides efficient support for the use of a limited set of control strategies.

Simulation studies were performed which evaluate alternative methods for scheduling tasks on parallel architectures. Two methods were examined which allow the OR-parallel execution of logic programs with no communication overhead.

A study was performed evaluating two alternative methods for incorporating integrity constraints into query processing in PRISM. In the first method, separate constraint processors are introduced which check constraints at run-time. In the second method, constraints are incorporated through compile-time transformations. The study indicates that constraints are useful in query processing and that the compile-time methodology results in more efficient performance than checking constraints at run-time.

In addition to the above, work continued in the area of informative answers to queries in deductive databases.

1. Introduction

This report constitues the final report of work performed on parallel problem solving and deductive databases under Air Force Grant AFOSR 90-0027 Under the grant experiments were performed using the PRISM system on the BBN Butterfly parallel architecture to evaluate message passing strategies and the use of constraints in problem solving. Several enchancements were made to the PRISM system including the design and implementation of two inference engines. Simulation studies were performed which evaluate methods for distributing tasks to processors which do not require interprocess communication or synchronization. In addition, we extended our work in cooperative answers to database queries.

Due to budget cuts several proposed investigations were not performed. These areas are:

1. AND-parallel execution of PRISM
2. Investigations in AI-Based parallel software
3. Coupling databases to processors in parallel/non-parallel environments.

As a consequence of the work accomplished during the past year of the grant period we have published:

1. 1 Ph.D. thesis with partial support from the grant [Giuliano90b].
2. 1 journal article [Chakravarthy90]
3. 3 book chapters [Giuliano90a,Gal,Grant].
4. 2 refereed conference papers [Lin,Liu]
5. 1 invited conference paper [Minker]
6. 1 workshop paper [Gaasterland90b]
7. 1 technical report [Gaasterland90a]

A list of papers that have been written during the current grant is provided in Section 2.3. A total of 7 published papers have appeared in print, and a total of 3 additional papers have been accepted for publication during the grant period. The references in Section 4 of the proposal list all papers produced with support from the AFOSR during the several years that the Air Force has been supporting the research.

2. Accomplishments on Effort During Period November 1989 – November 1990

In this section we report on the accomplishments obtained during the period from November 1989 to November 1990. This section is subdivided into three major parts. The first section, 2.1, describes the accomplished research with respect to parallel logic programming. The second section, 2.2, describes the accomplished research with respect to deductive databases. Section 2.3 contains a list of all papers and reports written during the grant period.

Three major efforts were proposed:

1. Investigations in parallel logic programming using PRISM
2. Investigations in AI–based parallel software
3. Investigations in deductive databases

Due to budget cuts the second research area was dropped as well as several sub-tasks of the first and third research areas. Each of the tasks, in which work was performed, are described in the following sections.

2.1. Investigations in Parallel Logic Programming Using Prism

There were five major tasks involving the parallel execution of logic programs that we planned to undertake during the current grant period.

1. *Continued experiments with PRISM*
2. *An evaluation of constraints in parallel problem solving*
3. *Stack based execution models*
4. *Memory management for the shared memory model of PRISM*
5. *Experiments with AND-parallelism*

Emphasis was placed on tasks in the first four areas. Substantial progress has been made in the four areas studied.

2.1.1. Continued Experiments with PRISM

In previous grant periods the PRISM system was implemented on the BBN Butterfly and McMOB parallel architectures. In this grant period experiments have been performed evaluating the PRISM system and substantial improvements have been made to the PRISM inference engine. Both theoretical and empirical studies were performed on task distribution in message passing parallel architectures.

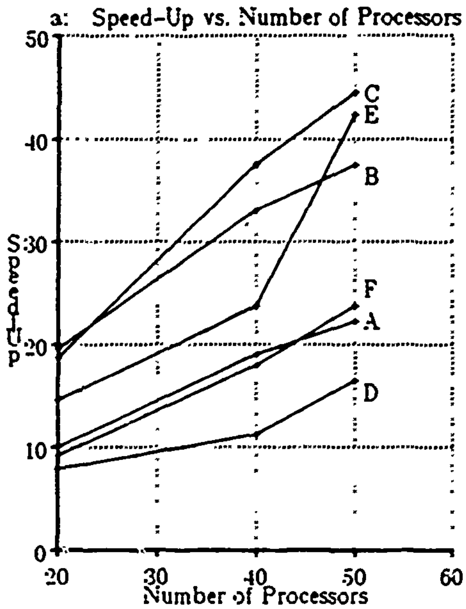
A new inference engine for PRISM has been designed and implemented. The new engine supports the same control features as the old engine and executes ten to twenty times faster. The engine achieves more efficient execution through the use of binding environments for variables. The current implementation is operational on sequential architectures as a stand alone logic programming system. A methodology for achieving parallel execution has been designed for the new inference engine and is currently being implemented.

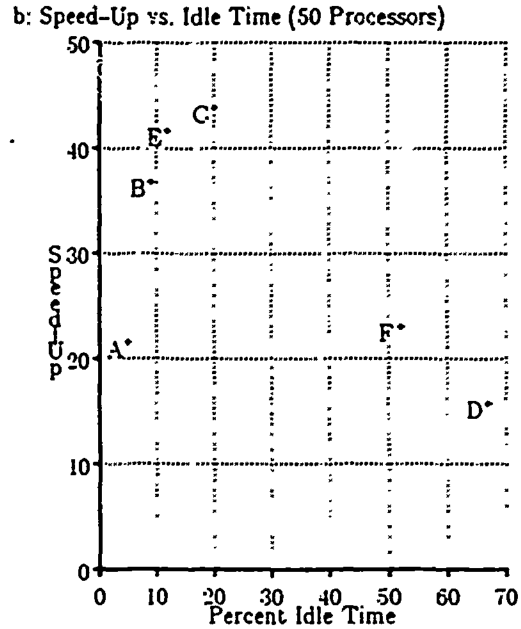
Message passing protocols for distributing tasks to processors in PRISM have been evaluated [Giuliano90a]. In PRISM, problem solving machines (PSMs) with extra work actively seek out idle processors by sending messages. If no machines are available when a resource allocation message is sent, then an overhead is incurred for no increase in parallel performance. Alternative protocols are investigated for knowing when resources are available for parallel problem solving. The protocols trade off the number of failed resource allocations for the amount of machine utilization. Six protocols (A – F) were evaluated in the experiments. In protocol A processors attempt to allocate remote resources regardless of failed resource allocations. In protocol D processors attempt to allocate remote resources until a failed resource allocation occurs (a readback). In the remaining protocols (B,C,E, and F) processors communicate processor availability. The protocols differ in when they send available messages and the addressing mode used to send the message (see the table in figure 1). Three benchmark programs were executed on the BBN Butterfly version of PRISM using the alternative message passing protocols. Experimental results are given in Figures 1a–1c for a version of the 8-queens program. Figure 1a shows the speed-up obtained over execution on a single processor for each of the protocols. Figure 1b shows the idle time in the system versus the speed-up when executing with 50 processors. Figure 1c shows the number of readbacks (i.e. failed processor allocations) versus the speed-up when executing with 50 processors. The figures show a clear tradeoff between idle time and failed resource allocations. Protocols with low idle time have a high number of failed resource allocations. Protocols with high idle time have a low number of failed resource allocations. The best performance occurs in protocol C which balances the number of failed resource allocations and the idle time.

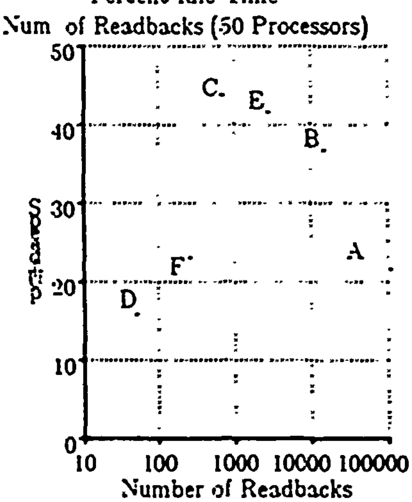
Another study evaluates the use of task scheduling strategies to improve the parallel performance of PRISM [Giuliano90a.Giuliano90b]. The study compares breadth-first versus depth-first task distribution strategies. In breadth-first execution tasks are distributed to new processors in a breadth-first manner and tasks are executed locally in a depth-first manner. In depth-first execution both remote and local tasks are scheduled in a depth-first manner. Four benchmark programs were executed on the BBN Butterfly version of PRISM using the alternative distribution strategies. Experimental results are given in Figure 2 for a version of the 8-queens program. Two tables are given. The first table shows the speed-up versus the number of processors for each protocol. The second table shows the number of task invocations versus the number of processors for each protocol. The study indicates that breadth-first scheduling results in better parallel performance by decreasing the number of tasks distributed among the processing elements.

Theoretical analysis and simulation studies were performed on the distribution of OR-parallel tasks to processors. A study evaluated an approach for distributing tasks to processors which does not involve communication between processing elements. In *random scheduling* [Janakarim] each processor is initialized with the same task and performs a depth-first search on the task. The processors search different portions of the proof tree by randomly ordering the program clauses to be executed. The idea is that some processors will select a short path and will thus find the solution quickly. Random scheduling was evaluated using simulation studies and theoretical analysis [Lin]. The evaluation demonstrates that on the average. random scheduling is not worse than approaches which use interprocess communication even if communication costs are assumed to be zero. With realistic communication costs. random scheduling performs better than approaches which require communication. Random scheduling is not efficient for problems in which all solutions are required. If all solutions are required, then each processor will randomly explore the entire search space.

A modification of random scheduling is *rule-based scheduling*. Like random scheduling, rule based scheduling does not entail communication between the parallel processing elements. All processors are initialized with the same tasks and perform a depth-first traversal of the search space. In rule-based scheduling, processors order the







$$Speed\text{-}up = \frac{Runtime}{Time\ on\ Single\ Processor}$$

$$Idle\ Time = \frac{Sum\ of\ Idle\ Time\ \text{for}\ each\ Processor}{Number\ of\ Processors \times Runtime}$$

Communication	Condition	Location Sent	Name
Processors Available	PSM becomes free	all busy PSM	B
		single busy PSM	C
	PSM becomes free after readback	all busy PSM	E
		single busy PSM	F
None	Stop sending queries after readback.	-	D
	Send queries regardless of readbacks	-	A

Message Passing Protocols

Figure 1: Message Passing Protocols for the 8 Queens on BBN Butterfly

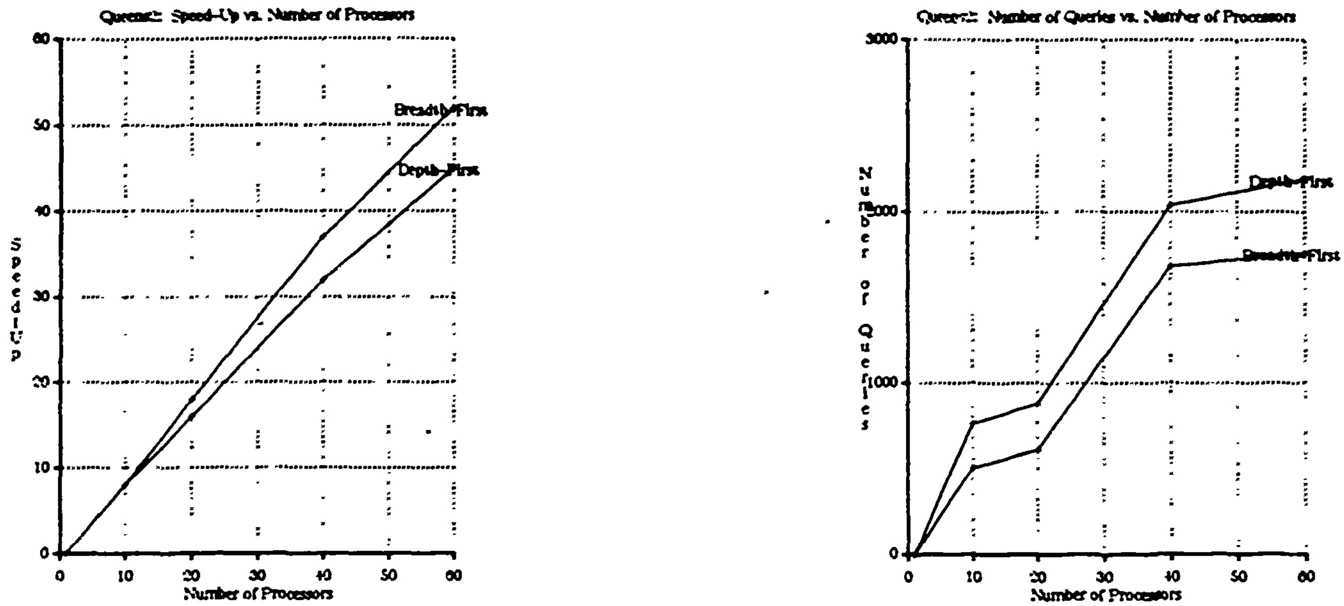


Figure 2: Queens2 Program: Breadth-First vs Depth-First Node Distribution

clauses at choice points based upon simple rules. For example, consider a program which has two alternative branches at each level of the proof tree. In this case, a rule could specify that at each alternative branch the processors are divided in half. In this manner. processors are guaranteed to examine different portions of the proof tree. A backtracking search occurs once a branch has only a single processor searching it. This approach has no inter-process communication and allows multiple answers to be obtained in an efficient manner. Preliminary results obtained from simulation studies are promising. High speed-ups were obtained for programs with regular shaped execution trees. For example. the simulation shows that for the 8-queens program, a speed-up of 40 using 128 processors can be achieved. In contrast, the best concrete implementations. which use communication between processors, saturate with a speed-up of 14 after using 32 processors. Application programs with irregular shaped proof trees did not achieve good parallel speed-up. An initial investigation suggests that programs whose proof trees are irregular can be transformed so that their proof trees become regular.

Random and rule-based scheduling differ from scheduling techniques which require interprocess communication. In systems which require communication, an initial query is given to a single processor. The processor executes the task and distributes sub-tasks as they occur to new machines. Likewise, the other machines execute their tasks possibly distributing new sub-tasks to other processors. Interprocess communication and synchronization is required to distribute the tasks. In contrast, in rule or random-based scheduling, there is no overhead for distributing tasks to processors. Because random and rule-based scheduling do not require communication, the simulation techniques used in the evaluations are highly accurate. Thus, we expect that concrete implementations of the scheduling strategies will result in similar performance.

2.1.2. Constraints in Parallel Problem Solving

A series of experiments were performed evaluating two methodologies for incorporating constraints in PRISM [Gaasterland90a]. Constraints provide the ability to prune the search space of a program based upon semantic information. The first approach incorporates constraints at run-time by introducing constraint solving processors [Kohli]. In this approach deductive processors interact with constraint processors to detect proof states which violate the integrity constraints. The second approach incorporates constraints by compile-time transformations which intergrate constraints with the program clauses [Chakravarthy]. The resulting program may then be executed using only deductive processors. An evaluation of the two methodologies using the PRISM system on the BBN Butterfly parallel architecture yields the following results:

1. The evaluation demonstrates that widely used classes of logic programs can use constraints to improve query processing.
2. The evaluation demonstrates that the compile-time approach to incorporating constraints results in better performance than the run-time approach.
3. The evaluation reveals additional issues in the compile-time and run-time approaches which must be addressed to utilize the theories of [Kohli] and [Chakravarthy].

Application programs were encoded from three logic programming domains. The domains involve inheritance hierarchies, the use of clustered data, and generate-and-test programs. The evaluation shows that for each of the domains both methods for incorporating constraints performs better than not utilizing constraints. The runs incorporating the compile-time approach resulted in better performance than the run-time approach for all the program domains.

Experimental data is given for incorporating constraints in a inheritance hierarchy program on the BBN Butterfly version of PRISM. Experiments were performed in three modes without constraints, with constraint machines, and with constraints compiled into the source program. The experiments incorporated 1,2,4, and 6 processing elements. In the experiments incorporating separate constraint machines, each constraint machine is tightly coupled with a deductive processor. The results are given in Figure 3. The table shows that both the run-time and compile-time method for incorporating constraints performs better than not using constraints. In addition, executions using the compile-time method perform better than using the run-time method.

2.1.3. Stack-Based Execution Models

A new binding scheme which supports OR-parallel execution of logic programs on distributed memory architectures was designed and implemented. The scheme is designed for stack-based inference engines and is

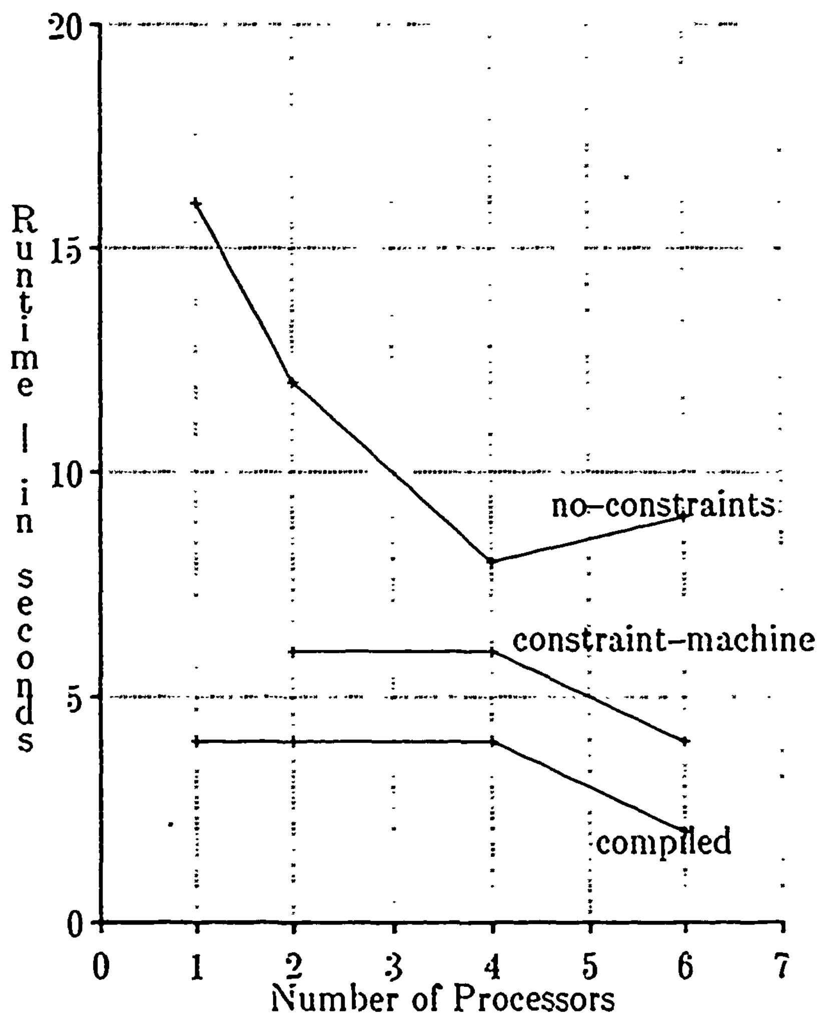


Figure 3: Run-Time Data for a Semantic Network Program

currently implemented in an inference engine for the PRISM problem solving component. The main idea is that on creating a new task enough information is packaged from the current processor such that the processors can execute independently. The advantage of the scheme is that it allows processors to execute deduction steps with no overhead over sequential execution. As such any improvements gained due to parallel execution offer improvements over execution on sequential architectures.

An evaluation of the binding scheme was performed using 60 processors on the BBN Butterfly version of PRISM [Giuliano90b]. The evaluation shows that programs in the benchmark set achieve from 5-30 factors of speed-up over sequential execution when using 60 processing elements (see Figure 4). Detailed statistics collected from the experiments indicate that many of the benchmark programs achieve low speed-up due to a high start-up cost for creating new parallel tasks. This is illustrated in Figure 5 which plots the percentage of time spent performing deduction steps versus the number of processors used in problem solving. The figure shows that for some benchmarks, processors spend up to 80 percent of their active time performing overhead for parallel execution.

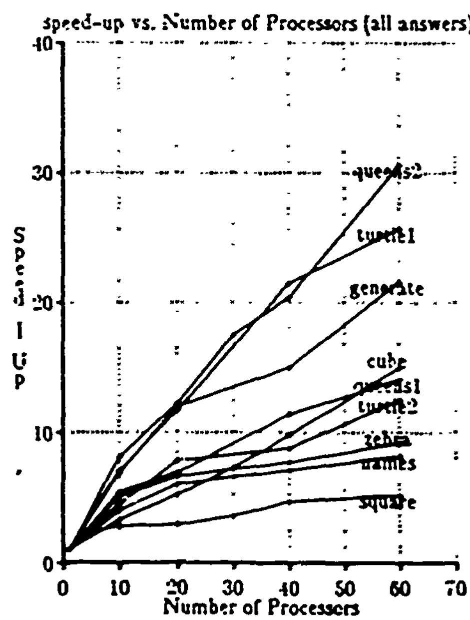


Figure 4: Speed Up in PRISM

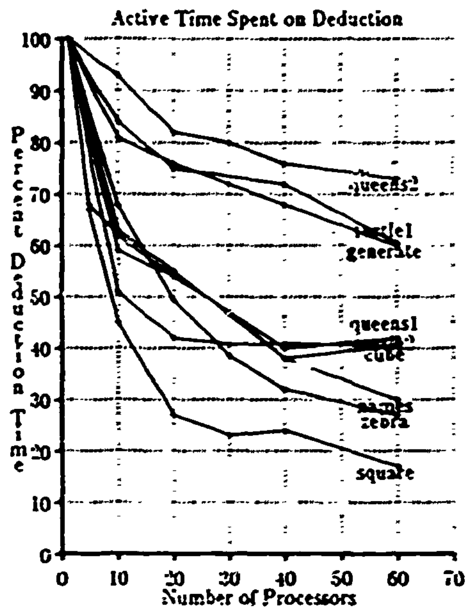


Figure 5: Percentage of Active Time Spent on Deduction

The use of task distribution strategies were investigated to lower the overhead for creating parallel tasks. The study shows that task distribution strategies can improve parallel performance by up to 50 percent by preventing trivial goals from being executed in parallel. Both run-time and compile-time strategies were shown to be useful for preventing trivial tasks from being executed in parallel.

2.1.4. Memory management for the shared memory version of PRISM

Although some work was performed on this task, no substantial results were obtained as the 128 node BBN Butterfly at our site was dismantled.

2.2. Investigations in Deductive Databases

The proposed investigations in this area were to:

1. Extend work on cooperative answers
2. Compute answers in non-Horn theories

3. Couple databases to processors in parallel/non-parallel environments

Emphasis was placed on the first and second area.

2.2.1. Extending Work on Cooperative Answers

The theory of cooperative answers developed by Gal and Minker [Gal] with support from the AFOSR works on both relational databases and on definite, nonrecursive, function free deductive databases. Until now, the focus of the cooperative answering effort was to collect information that could be included in a cooperative answer and to select salient information from the available pool. The theory takes advantage of integrity constraint information already available in a database to correct user misconceptions and to expand or to narrow the scope of answers. As of the previous grant period, the cooperative theory was fully implemented and the working system provided the basis for further research.

In the current grant period, our work focused on the following:

(1) designing a more sophisticated natural language output component;

(2) focusing the answer to address the original query and the user's needs;

(3) using information in the query's proof tree to provide additional cooperative information.

Even when the contents of an answer satisfy the query, address presuppositions, and are suitable to the user's purposes, expressing the answer well in natural language remains a complex task. Although it is easy to map predicates into natural language, the resulting language can be haphazard and flat. For example, if the clause to be realized in natural language is *grandmother(rose, sally) <--mother(rose,jack),father(jack,sally)*, a straightforward mapping would be "Rose is the grandmother of Sally because Rose is the mother of Jack and Jack is the father of Sally". Similarly, *mother(rose,jack) and mother(rose,george)* would become "Rose is the mother of Jack and Rose is the mother of George". Better responses would be "Rose is Sally's grandmother because she is the mother of Jack, who is Sally's father" and "Rose is the mother of Jack and George".

We have designed and partially implemented a theory to organize the elements of a cooperative response. The theory incorporates principles of coordination, subordination, anaphora, pronominalization, and discourse focus to organize the content of the answer. A phrase level grammar is then called to realize the individual phrases of the answer, and the final output is a more coherent, structured multisentential output. Our approach extends the representation given by Kowalski [Kowalski] which expresses n-ary relations as binary relations to incorporate information about grammatical categories. Dependencies across literals in the logical cooperative answer can be detected easily in the binary form, and they are used to identify opportunities for coordination, subordination and anaphora.

A prototype system has been implemented which transforms the cooperative response into the binary relations containing grammatical category information, collects dependencies across the binary relations, and uses them to organize the natural language output. Right now, the system can produce all allowable organizations. Further work must be done to select among organizations.

Toward developing criteria for organization, we have examined how to use the information gathered in translating the natural language query into a logical query and then into the database language query. Although the relations in the database query capture the intention of the user's natural language query, their literal meaning can differ. For example, a database relation may contain arguments that the user doesn't care about. Any values that those arguments obtain during the answering process should be suppressed in the natural language output unless they are determined to be of incidental use to the user.

Our prototype system uses information gathered in the translation from natural language to database language to focus the cooperative response. The resulting cooperative answers address the actual query asked by the user rather than the database version.

2.2.2. Computing Answers in Non-Horn Theories and Null Values

An execution model was designed implementing SLINF resolution for disjunctive logic programs. The model extends the execution model for PROLOG to allow disjunctive clauses. The implementation is in an initial phase. Currently a loader has been designed and tested for disjunctive clauses. The inference engine itself has not been completed.

2.3. Papers and Reports Published from November 1989 – November 1990

Below we list the papers or reports written during the period November 1989- November 1990. As a consequence of the work, we have published 10 papers.

A list of all papers and reports written since the AFOSR first sponsored the work is given in the bibliography in Section 4. The list of papers and reports that have been published during the present grant period are:

Bibliography

1. Chakravarthy, U.S., Grant, J. and Minker, J. "Logic Based Approach to Semantic Query Optrimization", *ACM Transactions on Database Systems, Vol. 15, No. 2,* June 1990, 162–207.
2. Gaasterland, T., Giuliano, M., Litcher, A., Liu, Y., & Minker, J. "Using Integrity Constraints to Control Search in Knowledge Base Systems", UMIACS–90–27, TR–2416, Feb. 1990
3. Gaasterland, T., Minker, J., Rajasekar, A., "Knowledge Base Systems – A Deductive Database Approach", Workshop on Knowledge Base Management Systems, AAAI 90.
4. Gal, A., & Minker, J. "Producing Cooperative Answers in Deductive Databases" In: Logic and Logic Grammer for Language Processing (Eds. Patrick St. Dizier & Szpakowicz S.) L.S. Horward Ltd. (to appear).
5. Giuliano, M., "The Control and Execution of Parallel Logic Programs", Ph.D. Thesis, Department Of Computer Science, University of Maryland, College Park Md. (in preparation)
6. Giuliano, M., Kohli, M., Minker, J. and Durand, I. "PRISM: A Testbed for Parallel Control," In: *Parallel Algorithms-for Machine Intelligence,* L. Kanal, V. Kumar, P.G. Gopalakrishnan (Editors), Springer–Verlag 1990.
7. Grant, J. and Minker, J. "Integrity Constraints in Knowledge Based Systems" In: *Knowledge Engineering, Vol. II, Applications,* H. Adeli (Editor), McGraw–Hill Publishers, 1990, 1–25.
8. Minker, J. "Toward A Foundation of Disjunctive Logic Programming," (Invited Banquet Address), Proceedings of the North American Conference on Logic Programming, 1215–1235, Ewing L. Lusk and Ross A. Overbeek (Editors), MIT Press, 1989.
9. Lin, Z., "Expected Performance of the Randomized Parallel Backtracking Method", in Proceeedings of the 1989 North American Conference on Logic Programming, 677–696, Ewing L. Lusk and Ross A. Overbeek (Editors), MIT Press, 1989, pages 677–696.
10. Liu, Y., "Null Values in Definite Programs", To appear in 1990 North American Conference of Logic Programming.

Bibliography

[Chakravarthy87]
Chakravarthy, U.. Grant, J., Minker, J., Foundation of semantic query optimazation for deductive databases, in *Foundations of Deductive Databases and Logic Programming*, Ed. Minker, J., Morgan-Kaufmann, 1987.

[Chakravarthy90]
hakravarthy, U.S., Grant, J. and Minker, J. "Logic Based Approach to Semantic Query Optrimization", *ACM Transactions on Database Systems, Vol. 15, No. 2*, June 1990, 162–207.

[Cohen]
Cohen, J., "Constraint Logic Programming Languages", in *Communications of the ACM*, Vol. 33, No. 7, July 1990.

[fin88]
Kass, R., Finn, T., "Modeling the User in Natural Language Systems", in *Computational Linguistics*, Vol. 14, No. 3., September 1988, pages 5–22.

[fin87]
Finn, T., "GUMS – A General User Modelling Shell" TR MS-CIS-87-74 LINC LAB 80, University of Pennsylvania, Philadelphia, PA, 1987.

[Fu] Fu, Li-Min, "Combining Nueral and Knowledge Base Approaches to Artificial Intelligence", in *Methodologies for Intelligent Systems, 4*, Ed. Ras, Z., 1989.

[Gaasterland90a]
Gaasterland, T., Giuliano, M., Litcher, A., Liu, Y., & Minker, J. "Using Integrity Constraints to Control Search in Knowledge Base Systems", UMIACS-90-27, TR-2416, Feb. 1990.

[Gaasterland90b]
Gaasterland, T., Minker, J., Rajasekar, A., "Knowledge Base Systems – A Deductive Database Approach", Workshop on Knowledge Base Management Systems, AAAI 90.

[Gal] Gal, A., & Minker, J. "Producing Cooperative Answers in Deductive Databases" In: Logic and Logic Grammer for Language Processing (Eds. Patrick St. Dizier & Szpakowicz S.) L.S. Horward Ltd. (to appear).

[Giuliano90a]
Giuliano, M., Kohli, M., Minker, J. and Durand, I. "PRISM: A Testbed for Parallel Control," In: *Parallel Algorithms for Machine Intelligence*, L. Kanal, V. Kumar, P.G. Gopalakrishnan (Editors), Springer-Verlag 1990.

[Giuliano90b]
Giuliano, M., "The Control and Execution of Parallel Logic Programs", Ph.D. Thesis, Department Of Computer Science, University of Maryland, College Park Md.

[Grant]
Grant, J. and Minker, J. "Integrity Constraints in Knowledge Based Systems" In: *Knowledge Engineering, Vol. II, Applications*, H. Adeli (Editor), McGraw-Hill Publishers, 1990, 1–25.

[Janakarim]
Janakiram, V. , Agrawal, D., and Mehrotra, R., "A Randomized Parallel Backtracking Algorithm", in *IEEE Transactions on Computers*, Vol. 37, NO. 12, December 1988.

[Kale]
Kale, L.V., "Parallel Execution of Logic Programs: The REDUCE-OR Process Model", In *Proc. of the Fourth International Conference on Logic Programming*, May 1987.

[Kohli83]
Kohli, M., Minker, J., "Control in Logic Programs Using Integrity Constraints", In *Proc. of the 1983 Logic Programming Workshop*, Algarve, Portugal, June 1983, pp 153–170.

[Kowalski]

Kowalski, R.A., "Logic for Problem Solving", Elsevier North Holland Inc., New York, 1979.

[Kumar87a]

Kumar, V., Nageshwara, R., "Parallel Depth First Search. Part I. Implementation", in International Journal of Parallel Programming, Vol. 16, No. 6, 1987.

[Kumar87b]

Kumar, V., Nageshwara, R., "Parallel Depth First Search. Part II. Analysis", in International Journal of Parallel Programming, Vol. 16, No. 6, 1987.

[Lin] Lin, Z., "Expected Performance of the Randomized Parallel Backtracking Method", in Proceeedings of the 1989 North American Conference on Logic Programming, 677–696, Ewing L. Lusk and Ross A. Overbeek (Editors), MIT Press, 1989, pages 677–696.

[Liu] Liu, Y., "Null Values in Definite Programs", To appear in 1990 North American Conference of Logic Programming.

[Lusk]

Lusk, E., et. al., "The Aurora OR-Parallel PROLOG System", *Proceedings of the International Conference on Fifth Generation Computer Systems*, ICOT, 1988.

[Minker]

Minker, J. "Toward A Foundation of Disjunctive Logic Programming," (Invited Banquet Address), Proceedings of the North American Conference on Logic Programming, 1215–1235, Ewing L. Lusk and Ross A. Overbeek (Editors), MIT Press, 1989.

[Moko-Oka]

Moko-Oka, T., "Challenge for Knowledge Infromation Processing Systems (Preliminary Report on Fifth Generation Computer Systems)", in *Proceedings of the International Conference on Fifth Generation Computer Systems*, ICOT, 1981, pgs. 1–85.